เทคนิคพัฒนาการ
ภาษาไทย
บนเครื่อง ไมโครคอมพิวเตอร์

ตอน4

รหัสภาษาไทย กับกรรมวิธีการประมวลผล การจัดระดับ และการจัดเรียงคำไทย

ยืน ภู่วรวรรณ

เกริ่นนำ

ผู้เขียนได้กล่าวถึงเทคนิคพัฒนาการพื้นฐานไปแล้วสามตอน ซึ่งเป็นเรื่องพื้นฐานที่เกี่ยวข้องกับการพัฒนาการประยุกต์ใช้ภาษาไทยกับคอมพิวเตอร์ ผู้เขียนและคณะแห่งห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ได้ทำการวิจัยในเรื่องภาษาไทยกับคอมพิวเตอร์ไว้หลายเรื่อง บางเรื่องก็เคยเสนอให้กับคณะกรรมการวิจัยแห่งชาติแล้วตั้งแต่ปี พ.ศ. 2525, 2526 บางเรื่องก็เสนอในที่ประชุมวิชาการที่ต่างๆ สำหรับในวารสารไมโครคอมพิวเตอร์นี้ ผู้เขียนตั้งใจจะนำเอาผลงานวิจัยดังกล่าวมาย่อยและขยายความเป็นเรื่องๆ ไป โดยเน้นในเรื่องความเข้าใจที่ง่าย ผู้เขียนจึงขอตัดเอาส่วนของการพิสูจน์ทางคณิตศาสตร์และอัลกอริทึมบางอย่างออกไปก่อน โดยเฉพาะอย่างยิ่งในเรื่องของฮาร์ดแวร์ซึ่งยังมีขั้นตอนเฉพาะอยู่ ผู้เขียนจะให้เฉพาะหลักการไว้ก่อน ส่วนซอฟท์แวร์นั้นถ้านำมาลงก็อาจจะยาวมากจนเนื้อที่ของไมโครไม่สามารถให้ลงได้ แต่ก็จะย่อยและเขียนโปรแกรมให้เห็นการนำไปใช้ ผู้เขียนจึงขอสร้างความเข้าใจพื้นฐานไว้ก่อน และจะเพิ่มขยายความตามลำดับต่อไป

รหัสคอมพิวเตอร์

คอมพิวเตอร์เป็นเครื่องมือทางอิเล็ก–ทรอนิกส์ที่ทำงานตามรูปแบบของสัญญาณทางไฟฟ้าที่ป้อนให้ และเพื่อให้เกิดการทำงานเชิงตัวเลขจึงมีผู้แทนรหัส “0” และ “1” เพื่อแทนระดับของสัญญาณไฟฟ้าสองระดับ คอมพิวเตอร์จึงมีพื้นฐานการทำงานด้วยตัวเลข 2 ตัวคือ 0, 1 ที่เรียกว่า ไบนารี

ในภาษาอังกฤษ มีผู้กำหนดรหัสตัวอักขระภาษาอังกฤษไว้แล้ว โดยแต่เดิมรหัสเหล่านั้นมาจากฐานของบัตรเจาะรู และได้กำหนดรหัสเป็นรหัสที่ชื่อว่า EBCDIC ดัง

รูปที่ 1 และนิยมใช้กันมาตั้งแต่ยุคแรก

รหัส EBCDIC ยังคงมีใช้กันในเครื่องคอมพิวเตอร์ขนาดใหญ่ เช่น ไอบีเอ็ม, ยูนิ-แวค, เอ็นอีซี ฯลฯ รหัส EBCDIC จึงมีพื้นฐานที่เป็นที่เก็บข้อมูลจำนวนมหาศาลมาแล้ว และข้อมูลในฐานข้อมูลใหญ่ๆ เป็นจำนวนมากเก็บด้วยรหัส EBCDIC ถ้าดูอย่างผิวเผินจะพบว่าตารางรหัส EBCDIC มีการจัดวางตัวหนังสือที่ไม่เหมาะกับการจัดลำดับ แต่ EBCDIC ก็ยังมีข้อดีหลายๆ เรื่องและมิได้เป็นปัญหาทางด้านการจัดเรียงข้อมูลแต่อย่างไร รหัส EBCDIC จึงยังคงอยู่ได้และอยู่ต่อไปอีก เพราะมีบริษัทใหญ่ใช้กัน

องค์การว่าด้วยมาตรฐานระหว่างประเทศ(ISO) ได้เล็งเห็นความสำคัญของการจัดรหัสมาตรฐาน จึงได้ทำการร่างรหัส ISO 646/1983 ดังรูปที่ 2 ซึ่งเป็นรูปแบบที่เครื่องไมโครคอมพิวเตอร์ใช้กันในปัจจุบันนี้ โดย ISO เน้นเป็นรหัสสำหรับการแลกเปลี่ยนข้อมูลและข่าวสาร

อย่างไรก็ตาม ISO ก็ยังเปิดช่องรหัสในส่วนตาราง 5B-5D และ 7B-7D เป็นรหัสที่ยอมให้แต่ละประเทศกำหนดกันขึ้นมาเอง ซึ่งหมายถึงว่า**รหัสที่ใช้ในการแลกเปลี่ยน**

รูปที่ 1 ตารางรหัส EBCDIC ที่ยังคงมีใช้กันมากในเครื่องคอมพิวเตอร์ขนาดใหญ่

รหัส EBCDIC สากล (แสดงในรูปของตาราง 16×16)

4 บิทต้น / 4 บิทท้าย	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø	NUL	DLE			SP	&	–						{	}	\	Ø
1	SOH	DC1				/			a	j	~		A	J		1
2	STX	DC2							b	k	s		B	K	S	2
3	ETX	TM							c	l	t		C	L	T	3
4	PF	RES							d	m	u		D	M	U	4
5	HT	NL							e	n	v		E	N	V	5
6	LC	BS							f	o	w		F	O	W	6
7	DEL	IL							g	p	x		G	P	X	7
8	GE	CAN							h	q	y		H	Q	Y	8
9	RLF	EM							i	r	z		I	R	Z	9
A	SMM	CC	SM		¢	!	¦	:								LVM
B	VT	CU1	CU2	CU3	.	$	,	#								
C	FF	IFS		DC4	<	*	%	@								
D	CR	IGS	ENQ	NAK	(	)	_	'								
E	SO	IRS	ACK		+	;	>	=								
F	SI	IUS	BEL	SUE	|	¬	?	"								EO

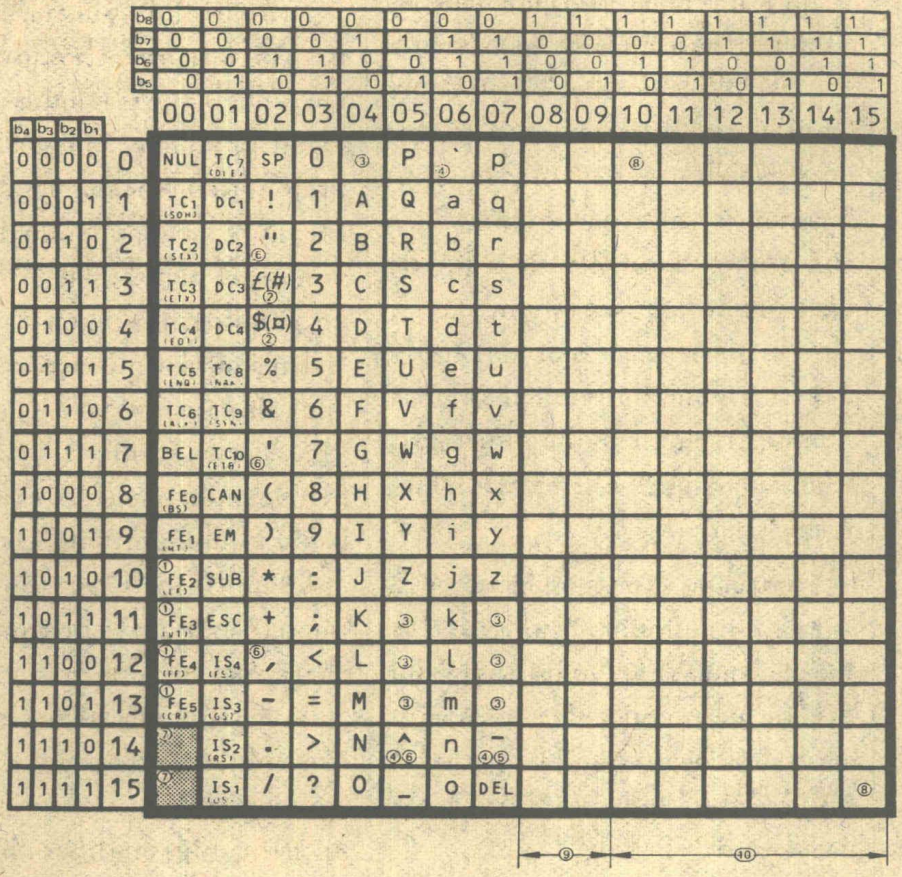
คำย่อใช้เรียกรหัสควบคุม

วิธีการอ่านตาราง

ตัวอย่างเช่น รหัส W : อยู่ที่บรรทัดที่ 6 คอลัมน์ที่ A

เพราะฉะนั้น รหัส W คือ A6 (hex) หรือ 1010 0110 (binary)

รูปที่ 2 ตารางรหัสตามมาตรฐาน IS0646/ 1983



b4 b3 b2 b1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 0 0 0 0	NUL	TC7 (DLE)	SP	0	③	P	`	p			⑧					
0 0 0 1 1	TC1 (SOH)	DC1	!	1	A	Q	a	q								
0 0 1 0 2	TC2 (STX)	DC2	" ⑤	2	B	R	b	r								
0 0 1 1 3	TC3 (ETX)	DC3	£(#) ②	3	C	S	c	s								
0 1 0 0 4	TC4 (EOT)	DC4	$(¤) ②	4	D	T	d	t								
0 1 0 1 5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u								
0 1 1 0 6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v								
0 1 1 1 7	BEL	TC10 (ETB) ⑥	'	7	G	W	g	w								
1 0 0 0 8	FE0 (BS)	CAN	(	8	H	X	h	x								
1 0 0 1 9	FE1 (HT)	EM	)	9	I	Y	i	y								
1 0 1 0 10	① FE2 (LF)	SUB	*	:	J	Z	j	z								
1 0 1 1 11	① FE3 (VT)	ESC	+	;	K	③	k	③								
1 1 0 0 12	① FE4 (FF)	IS4 (FS) ⑥	,	<	L	③	l	③								
1 1 0 1 13	① FE5 (CR)	IS3 (GS)	-	=	M	③	m	③								
1 1 1 0 14	⑪	IS2 (RS)	.	>	N	^ ④⑥	n	~ ④⑤								
1 1 1 1 15	⑪	IS1 (US)	/	?	O	_	o	DEL								⑧

⑨ ⑩

NOTES ON TABLE 1

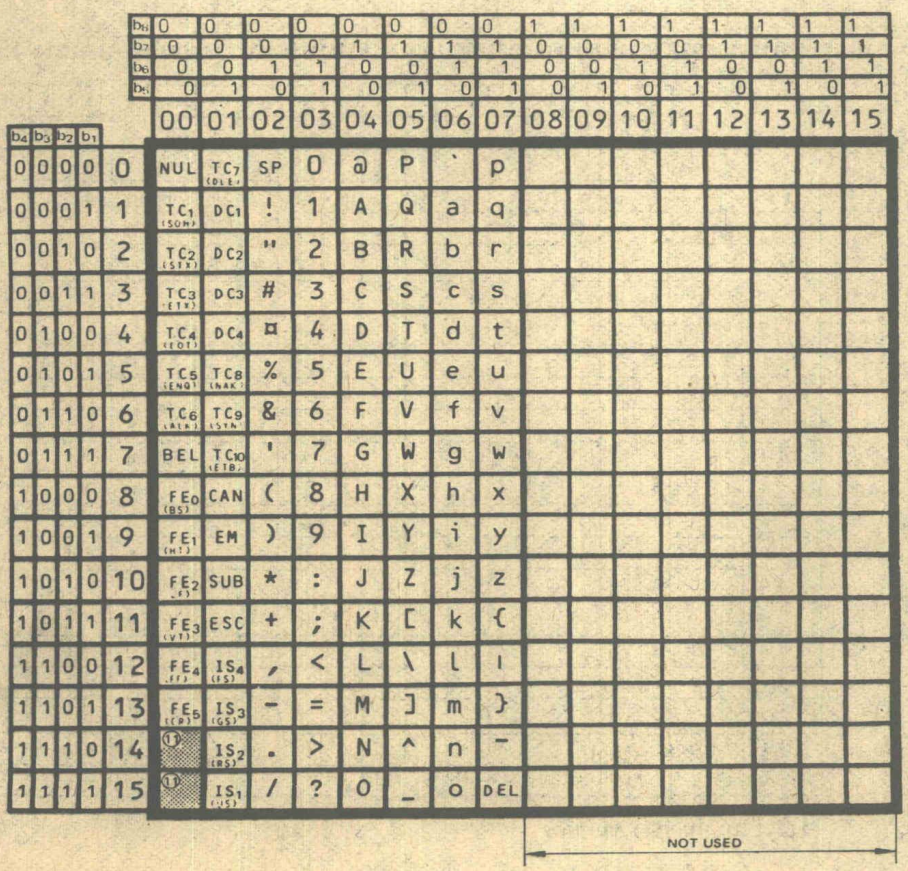
① The format effectors are intended for equipment in which horizontal and vertical movements are effected separately. If equipment requires the action of CARRIAGE RETURN to be combined with a vertical movement, the format effector for that vertical movement may be used to effect the combined movement. For example, if NEW LINE (symbol NL, equivalent to CR + LF) is required, FE_2 shall be used to represent it. This substitution requires agreement between the sender and the recipient of the data.

The use of these combined functions may be restricted for international transmission on general switched telecommunication networks (telegraph and telephone networks).

② The symbol £ is assigned to position 02/3 and the symbol $ is assigned to position 02/4. In a situation where there is no requirement for the symbol £, the symbol # (NUMBER SIGN) may be used in position 02/3. Where there is no requirement for the symbol $, the symbol ¤ (CURRENCY SIGN) may be used in position 02/4. The chosen allocations of symbols to these positions for international information interchange shall be agreed between the interested parties. It should be noted that, unless otherwise agreed between sender and recipient, the symbols £, $ or ¤ do not designate the currency of a specific country.

③ National use positions. The allocation of characters to these positions lies within the responsability of national standardization bodies. These positions are primarily intended for alphabet extensions. If they are not required for that purpose, they may be used for other graphics.

④ Positions 05/14, 06/0 and 07/14 provide for the characters UPWARD ARROW HEAD, GRAVE ACCENT and OVERLINE. However, these positions may be used for other graphic characters when it is necessary to have 8, 9 or 10 positions for national use.



b4 b3 b2 b1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 0 0 0 0	NUL	TC7 (DLE)	SP	0	@	P	`	p								
0 0 0 1 1	TC1 (SOH)	DC1	!	1	A	Q	a	q								
0 0 1 0 2	TC2 (STX)	DC2	"	2	B	R	b	r								
0 0 1 1 3	TC3 (ETX)	DC3	#	3	C	S	c	s								
0 1 0 0 4	TC4 (EOT)	DC4	¤	4	D	T	d	t								
0 1 0 1 5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u								
0 1 1 0 6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v								
0 1 1 1 7	BEL	TC10 (ETB)	'	7	G	W	g	w								
1 0 0 0 8	FE0 (BS)	CAN	(	8	H	X	h	x								
1 0 0 1 9	FE1 (HT)	EM	)	9	I	Y	i	y								
1 0 1 0 10	FE2 (LF)	SUB	*	:	J	Z	j	z								
1 0 1 1 11	FE3 (VT)	ESC	+	;	K	[	k	{								
1 1 0 0 12	FE4 (FF)	IS4 (FS)	,	<	L	\	l	|								
1 1 0 1 13	FE5 (CR)	IS3 (GS)	-	=	M	]	m	}								
1 1 1 0 14	⑪	IS2 (RS)	.	>	N	^	n	~								
1 1 1 1 15	⑪	IS1 (US)	/	?	O	_	o	DEL								

NOT USED

NOTES ON TABLE 1 (*continued*)

⑤ Position 07/14 is used for the graphic character ~ (OVERLINE), the graphic representation of which may vary according to national use to represent ~ (TILDE) or another diacritical sign, provided that there is no risk of confusion with another symbol included in the table.

⑥ The graphic characters in positions 02/2, 02/7, 02/12 and 05/14, have respectively the significance of QUOTATION MARK, APOSTROPHE, COMMA and UPWARD ARROW HEAD, however, these characters take on the significance of the diacritical signs DIAERESIS, ACUTE ACCENT, CEDILLA and CIRCUMFLEX ACCENT respectively when they are preceded or followed by the BACKSPACE character (00/8), (see also 5.2 of ISO 646).

⑦ No character is allocated to positions 00/14 and 00/15.

⑧ To ensure full compatibility with the code extension techniques described in ISO 2022, positions 10/0 and 15/15 are left undefined. The use of these two positions for data interchange requires agreement between the sender and the recipient of the data.

⑨ Positions for 08/0 to 09/15 are reserved for the allocation of a set of up to 32 control characters.

⑩ Positions from 10/1 to 15/14 are reserved for the allocation of a set of up to 94 graphic characters.

NOTE ON TABLE 2

⑪ No character is allocated to positions 00/14 and 00/15.

ข้อมูลข่าวสารที่ ISO กำหนดไม่ได้รวมส่วนของช่องเหล่านี้ไว้ แต่ก็มีผู้แนะนำและกำหนดเป็นรหัส ASCII สำหรับประเทศสหรัฐ–อเมริกา โดยนำรหัสตัวอักษร [\] ^{¦} และ ~ บรรจุลงในช่องว่างเหล่านั้น และก็ใช้กันมาจนดูคล้ายเป็นส่วนหนึ่งของ ASCII ไป

และด้านการเปิดช่องของการไม่รวมเข้าเป็นรหัสการแลกเปลี่ยนข้อมูลนี้เอง ผู้พัฒนาซอฟท์แวร์ที่ใช้ในการสื่อสาร โปรแกรมโอเอส หรือแม้แต่โปรแกรมประยุกต์ เช่น โลตัส 123 ก็ใช้รหัสเหล่านี้เป็นรหัสควบคุม โดยเฉพาะภาษาคอมพิวเตอร์ยุคใหม่ เช่น ซี โปรล็อก ฯลฯ ก็ใช้รหัสเหล่านี้เป็นรหัสควบคุมโครงสร้างของโปรแกรม และหากจะดูในเรื่องของโอเอส เช่น XENIX, MSDOS หรือ UNIX จะใช้รหัสเหล่านี้เป็นส่วนควบคุมการทำงานของโปรแกรมโอเอส เช่น ใช้ \ เป็น Path ของไดเรกทอรี ใช้ ¦ เป็นไพพ์ (pipe) เป็นต้น

จะเห็นได้ว่าช่องว่างเหล่านี้เป็นช่องทางที่มีความไม่แน่นอน เป็นช่องที่จะเปิดโอกาสให้เป็นรหัสควบคุมได้นอกเหนือจาก 00-2F แล้ว ดังนั้นผู้เขียนโปรโตคอลสื่อสารจึงนำเอารหัสส่วนนี้เป็นส่วนของการกำหนดเป็นโปรโตคอลบางอย่าง เช่น HDLC โปรโตคอลใช้บิท 1 ติดกัน 6 บิทเป็นตัวบอกการเริ่มต้นเฟรมข้อมูล ดังนั้นข้อมูลจะมีบิท 1 ติดกัน 6 บิทไม่ได้ ซึ่งจะมีกรรมวิธีการแทรก 0 เข้าไป

รหัสภาษาไทย

รหัสที่ใช้ในคอมพิวเตอร์นั้น แยกตามกลุ่มการใช้งานของระบบคอมพิวเตอร์ได้เป็น

1. Internal code คือรหัสที่ใช้ภายในเครื่อง เป็นรหัสที่จะเข้ามาอยู่ในหน่วยความจำของคอมพิวเตอร์ ใช้ในการประมวลผลด้วยซีพียู

2. Media code คือรหัสที่จะเก็บไว้ในตัวกลาง เช่น ดิสเกตต์ เทป รหัสนี้บางเครื่องไม่เหมือนกับ internal code เช่น การเก็บข้อมูลในดิสค์ของแอปเปิ้ลมีวิธีการถอดรหัส 6 เป็น 8 บิทตามวิธีการของแอปเปิ้ลเอง ดังนั้นฟอร์แมทของตัวกลางจึงอาจแตกต่างกันไป

3. Display code คือรหัสที่จะใช้สำหรับแสดงผลออกมาทางจอภาพ หรือทางเครื่องพิมพ์เพื่อความสวยงาม เช่น ภาษาไทยอาจมี 4 ระดับ แล้วลดระดับเหลือ 3 หรือ 2 ระดับ หรือบางทีใช้รหัส 2 ไบท์เพื่อลดระดับเหลือระดับเดียว เป็นต้น

4. Interchange code คือรหัสที่ใช้สำหรับการแลกเปลี่ยนข้อมูลและข่าวสาร เป็นรหัสที่ใช้แลกเปลี่ยนข้อมูลกันระหว่างระบบ ระหว่างอุปกรณ์ หรือระหว่างคอม–พิวเตอร์ด้วยกันเอง

หมายเหตุ ในทางอุดมคติเราอยากให้รหัสทั้ง 4 กลุ่มนี้เหมือนกันหมด เพราะจะไม่ต้องมีการแปลงไปมา แต่ในทางปฏิบัติเป็นไปได้ยาก เพราะความแตกต่างกันทั้งทางด้านเทคโนโลยีฮาร์ดแวร์และซอฟท์แวร์ของเครื่อง ดังนั้นเราจึงพบว่าคอมพิวเตอร์บางเครื่องใช้หลายรหัสได้

รหัสภาษาไทยของสมอ.

สำนักงานมาตรฐานผลิตภัณฑ์อุตสาห–กรรมได้ตั้งคณะกรรมการวิชาการศึกษารหัส และได้มีการกลั่นกรองจากผู้ทรงคุณวุฒิที่มาจากหน่วยงานภายนอกอีกหลายขั้นตอน จึงได้ประกาศร่างรหัสของสมอ.ออกมาดังรูปที่ 3 และรูปที่ 4 และจากขั้นตอนการดำเนินงานของสมอ.นั้น มีขั้นตอนที่กลั่นกรองโดยผู้ทรงคุณวุฒิที่ทำการศึกษาค้นคว้าทางด้านภาษาไทยกับคอมพิวเตอร์มาโดยเฉพาะ การกำหนดรหัสจึงต้องใช้เวลาเพราะหลังจากที่กรรมการวิชาการเสนอแล้ว ก็มีการเชิญบริษัทต่างๆ มาให้ข้อคิดเห็น เวียนให้บริษัทแสดงความคิดเห็น จากนั้นเสนอให้กับผู้ทรงคุณวุฒิทางคอมพิวเตอร์อื่นที่ไม่ใช่กรรมการวิชาการให้ความคิดเห็น แล้วจึงนำเข้าสู่การพิจารณาของคณะกรรมการสมอ.

อนึ่ง ขั้นตอนการเขียนร่างให้บริษัทฯ พิจารณานั้นได้รับการตอบกลับ 41 หน่วยงาน ผลปรากฏเห็นชอบตามร่างทุกประการ 21 หน่วยงาน, เห็นชอบตามร่างแต่มีข้อคิดเห็น

รูปที่ 3 ตารางรหัสมาตรฐานที่สมอ.กำหนดเป็นฉบับร่าง (รหัสบนตาราง ISO)

4 บิทต้น	b8	b7	b6	b5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4 บิทท้าย					0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b4	b3	b2	b1																	
0	0	0	0	0	0											ฐ	ภ	ะ	เ	๐
0	0	0	1	1	1										ก	ฑ	ม	ั	แ	๑
0	0	1	0	2	2										ข	ฒ	ย	า	โ	๒
0	0	1	1	3	3										ฃ	ณ	ร	ำ	ใ	๓
0	1	0	0	4	4										ค	ด	ฤ	ิ	ไ	๔
0	1	0	1	5	5										ฅ	ต	ล	ี	ๅ	๕
0	1	1	0	6	6										ฆ	ถ	ฦ	ึ	ๆ	๖
0	1	1	1	7	7										ง	ท	ว	ื	็	๗
1	0	0	0	8	8										จ	ธ	ศ	ุ	่	๘
1	0	0	1	9	9										ฉ	น	ษ	ู	้	๙
1	0	1	0	A	10										ช	บ	ส	ฺ	๊	ฯ
1	0	1	1	B	11										ซ	ป	ห	฿	๋	฿
1	1	0	0	C	12										ฌ	ผ	ฬ		์	
1	1	0	1	D	13										ญ	ฝ	อ		๐	
1	1	1	0	E	14										ฎ	พ	ฮ		๏	
1	1	1	1	F	15										ฏ	ฟ	ฯ	ๆ	๏	

☑ = เหมือน ISO 646 เดิม

รูปที่ 4 ตารางรหัสมาตรฐานที่สมอ.กำหนดเป็นฉบับร่าง (รหัส EBCDIC)

				4 บิทต้น																
					b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
					b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
					b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4 บิทท้าย						0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b4	b3	b2	b1			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0								B	⊙	ฯ	๛	๐				
0	0	0	1	1	1								๙				๑				
0	0	1	0	2	2					ก	จ	ฎ	ถ				๒				
0	0	1	1	3	3					ข	ฉ	ฐ	ท				๓				
0	1	0	0	4	4					ฃ	ช	ฑ	ธ				๔				
0	1	0	1	5	5					ค	ซ	ฒ	น				๕				
0	1	1	0	6	6					ฅ	ฌ	ณ	บ				๖				
0	1	1	1	7	7					ฆ	ญ	ด	ป				๗				
1	0	0	0	8	8					ง	ฎ	ต	ผ				๘				
1	0	0	1	9	9												๙				
1	0	1	0	A	10									ฝ	ร	ษ	ฯ		.	า	๋
1	0	1	1	B	11									พ	ฤ	ส	ะ	ี	เ	ๆ	์
1	1	0	0	C	12									ฟ	ล	ห	ั	ื	แ	็	๐
1	1	0	1	D	13									ภ	ฦ	ฬ	า	ื	โ	่	
1	1	1	0	E	14									ม	ว	อ	ำ	,	ใ	้	
1	1	1	1	F	15									ย	ศ	ฮ	ิ	๚	ไ	๊	

☒ = เหมือน EBCDIC เดิม

เพิ่มเติมอีกเล็กน้อย(ส่วนใหญ่เป็นเรื่องการใช้คำพูดในร่าง) 18 หน่วยงาน, มีหนึ่งรายเห็นชอบแต่มีข้อเสนอเพิ่มเติมให้ปรับปรุง และอีก 1 รายไม่เห็นชอบกับร่างหรืออาจสรุปได้ว่าเห็นด้วยกับสมอ. 40 ราย และไม่เห็นด้วย 1 ราย

จุดมุ่งหมายของรหัสสมอ. ตามที่ได้แบ่งแยกรหัสออกเป็น 4 แบบคือ internal, media, display และ interchange แล้ว รหัสของสมอ. มีเป้าหมายที่เด่นชัดตามขอบข่ายข้อ 1.2 ในร่าง(ดูร่างมาตรฐาน) โดยถือว่าเป็น**รหัส Interchange** คือเน้นการแลกเปลี่ยนข้อมูลและข่าวสาร ซึ่งแบ่งออกเป็น 2 แบบคือ แบบ ISO 646 และแบบ EBCDIC

โดยตัวอักษรไทยที่จะใช้**ในการแลกเปลี่ยนข้อมูลและข่าวสาร**ซึ่งกันและกัน ประกอบด้วยตัวอักษร 87 ตัวดังนี้

พยัญชนะ ได้แก่ ก ข ฃ ค ฅ ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ฤ ล ฦ ว ศ ษ ส ห ฬ อ ฮ

สระ ได้แก่ ็ ะ ั า ำ ิ ี ึ ื ุ ู เ แ โ ใ ไ ํ(นิคหิต) ฺ (พินทุ)

วรรณยุกต์และทัณฑฆาต ได้แก่ ่ ้ ๊ ๋ และ ์

เลขไทย ๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙

เครื่องหมายพิเศษทั่วไป ได้แก่ ๅ ๆ ฯ

เครื่องหมายพิเศษเฉพาะ B(บาท) ๙ (ยามักการ) ◎ ฟองมัน ฯ। (อันคั่นคู่) และ ๛ (โคมูตร)

ลักษณะของรหัสสมอ. ในกลุ่ม 646

รหัสที่เกี่ยวข้องกับไมโครคอมพิวเตอร์จะเกี่ยวพันโดยตรงกับรหัส 646 หรือตาราง ASCII คราวนี้มาดูว่าตัวอักษร 87 ตัวจะลงไปยังตารางได้อย่างไร โดยปกติเราจะต้องมองอนาคต(อย่าดูที่ปัจจุบันหรืออดีต) จากบทความเรื่อง NAPLPS:A New Standard for Text and Graphics โดย Jim Fleming และคณะเขียนลงในวารสารไบท์ ฉบับเดือน FEB 1983 Vol 8 No 2 หน้า 203 ถึง 254 และมีต่อในเล่ม MAR และ APR ซึ่งให้ข้อคิดเห็นของอนาคตในการจัดวางมาตรฐานรหัส North American Presentation Level Protocal Syntax ซึ่งจะใช้เป็นรหัสสำหรับการส่งเอกสาร วิดีโอเท็กซ์ซึ่งมีทั้งตัวหนังสือและข้อความ จากบทความดังกล่าวได้มองรหัส 7 บิท แล้วขยายตารางเพิ่มต่ออีกทีละ 128 ตัวอักษรออกมาเป็นกลุ่ม GR, G0, G1, G2, G3 ..... ดังรูปที่ 5 โดยมีวิธีการมองสำหรับอนาคตที่ได้เริ่มนำมาใช้แล้วที่อเมริกาเหนือในเรื่องวิดีโอเท็กซ์

โครงสร้างของ NAPLPS ยังได้รับการออกแบบการขยายรหัสต่อออกโดยใช้หลักการพื้นฐานของ ISO 2022.2 ซึ่งก็ถือว่าเป็นมาตรฐานระหว่างประเทศ

รหัสภาษาไทยจึงตั้งอยู่บนพื้นฐานของการเพิ่มต่ออีก 1 เพจหรือ 128 ตัวอักษร เพิ่มต่อจาก ASCII หรือ ISO 646 ซึ่งก็น่าจะตามแบบอย่าง เพื่อว่าอนาคตเราอาจจะมีการส่งวิดีโอเท็กซ์กันได้ จากหน้า 198 วารสารไบท์ ฉบับเดือน APR 1983 ได้แสดงวิธีการใช้รหัสโดยแบ่งรหัสใน 2 คอลัมน์แรกเป็นรหัสควบคุม โดยมีการเข้ารหัสคำสั่งที่ส่งเพื่อเป็นรูปภาพได้ รหัสในแต่ละเพจที่เหลืออีก 96 ตัวไว้สำหรับเป็นข้อมูล โดยมีวิธีการใช้เช่น กำหนดเป็นมาโครโปรแกรมของภาพแบบต่างๆ ได้ด้วยการสื่อสารข้อมูลเป็นสำคัญ เช่น

```
Define Macro 26
:
1   033 ESC ; รหัสเอสเคป
2   100 DFM ; บอกการกำหนดมาโคร
3   072 M26 ; ชื่อมาโคร
:
4   016 SO ; Select G1 (graphic)
.
.
.
ส่งรหัสกราฟิก
.
.
.
```

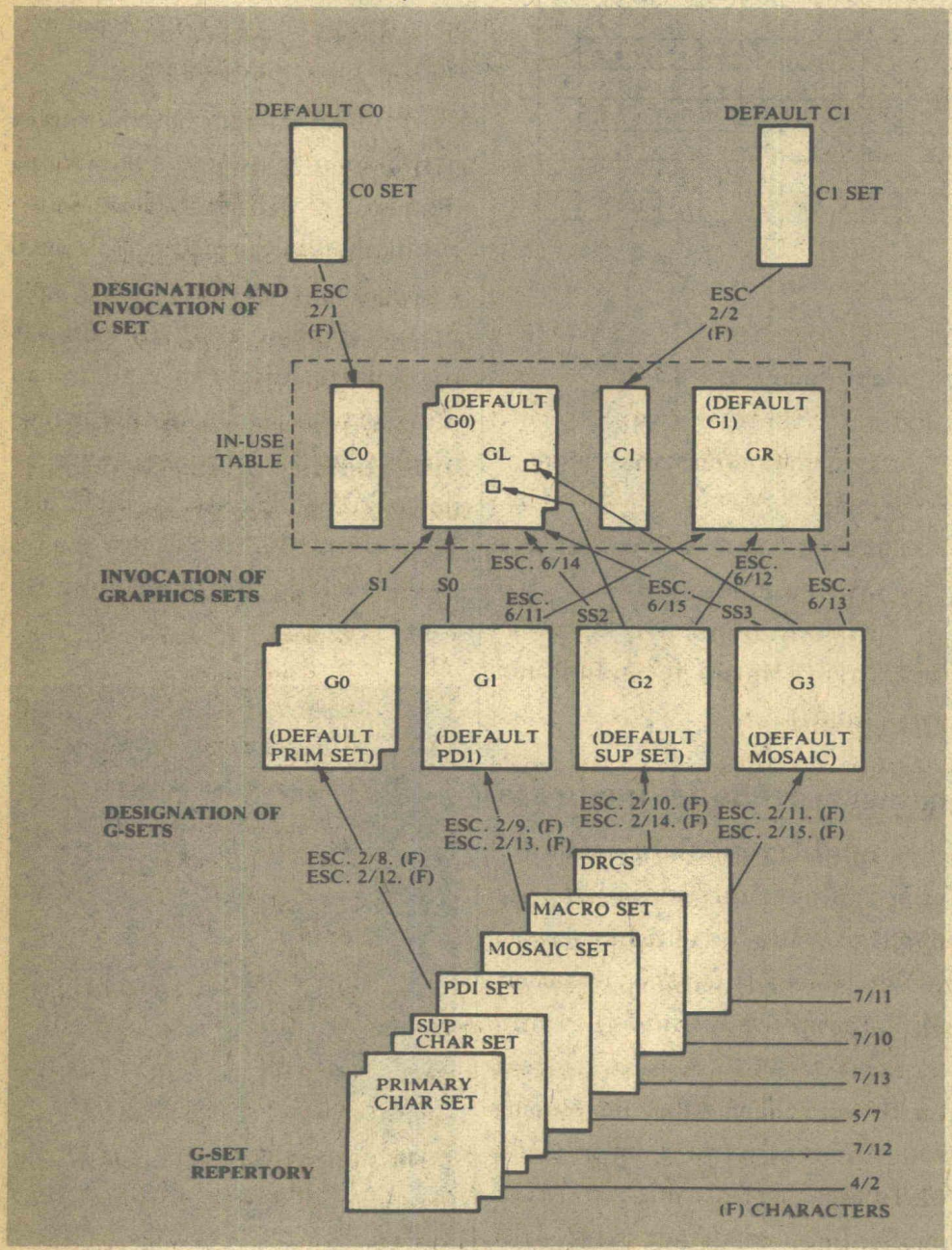
```
017 SI ; Select G0 (Text)
.
.
.
ส่งรหัสตัวอักษร
.
.
.
```

และจากระบบการสื่อสารข้อมูลที่กำหนด 7 layer Protocal ของ ISO **การส่งข้อมูลในระดับ physical layer กำหนดการส่ง 7 บิท** โดยตัวรหัสจะผ่าน level 1-5 โดยกันบิท 8 ไว้สำหรับเป็นพาริตี้ (ดูหน้า 216 ในไบท์ เล่ม Feb 1983)

ด้วยวิธีการนี้ เราจึงสามารถเผื่อให้กับเครื่องคอมพิวเตอร์ที่ใช้ 7 บิทจริงๆ ได้ (ปัจจุบันมีคอมพิวเตอร์หลายบริษัทในเมืองไทยใช้รหัสเพียง 7 บิท) ทำให้คอมพิวเตอร์เหล่านั้นใช้ภาษาไทยได้ จริงอยู่ไมโครคอมพิวเตอร์อาจใช้รหัส 8 บิทได้สบาย แต่มินิคอมพิวเตอร์หลายเครื่องใช้ไม่ได้ เมื่อเป็นเช่นนี้ตารางภาษาไทยจึงคล้ายกับแบบของ NAPLPS ที่สามารถเลือก G0, G1....GN ได้ โดยเมื่อภาษาไทยจะใช้ร่วมกับวิดีโอเท็กซ์ต่อไปก็น่าจะใช้ได้

ด้วยเหตุนี้เองรหัสภาษาไทยที่เน้นในเรื่องของ interchange code จะต้องมองเสมือนการออฟหรือออนบิท 8 เพื่อความคล่องตัวมากขึ้น

รูปที่ 5 โครงสร้างการขยายรหัสตาม NAPLPS (ดูรายละเอียดคำอธิบายเพิ่มเติมจากหนังสือไบท์)



จะจัดลงอย่างไร โดยมี 96 ช่อง ตัวอักษร 87 ตัว

ในแต่ละเพจมี 128 ช่อง แต่ต้องกันช่อง 2 คอลัมน์แรกซึ่งจะต้องถูกกำหนดเป็นรหัสควบคุมอย่างแน่นอน(ถ้าคิดแบบ NAPLPS) ดังนั้นจึงเหลือ 96 ช่อง แต่มีตัวอักษร 87 ตัว คณะกรรมการวิชาการสมอ. เห็นความสำคัญในการที่จะนำไปใช้ในอนาคตได้อย่างไม่มีปัญหา จึงต้องพิจารณาด้วยหลักการทางวิชาการดังนี้

1. การจัดลงจะต้องง่ายต่อการจัดระดับ
2. การจัดลงต้องง่ายต่อการจัดเรียงคำ
3. การจัดลงจะต้องใช้ได้ตามจุดมุ่งหมายของ Interchange code และพอจะเป็นฐานสากลที่ไม่ทำให้เกิดปัญหาในอนาคต

สำหรับในประเด็นข้อ 1 และข้อ 2 ผู้เขียนใคร่ให้เห็นอัลกอริทึมของผู้เขียน ที่เคยเสนอผลงานให้กับสภาวิจัยแห่งชาติมาแล้วซึ่งจะกล่าวในรายละเอียดตอนหลัง

การลดปัญหาที่อาจจะเกิดขึ้นในอนาคต

หากจะพิจารณาถึงข้อมูลโดยทั่วไปแล้ว ตัวอักษร 87 ตัวที่จะลงใน 96 ช่อง ซึ่งมีที่เหลือ 9 ช่อง และเพื่อลดความเสี่ยงในอนาคตน่าจะเว้นช่องในส่วน DB-DE และ FB-FE และเว้นสำหรับ space คือ A0 และ DEL คือ FF ไว้ จะเห็นว่า ส่วนของ A0 และ FF

จะต้องเว้นอย่างแน่นอน ดังนั้นตาม ISO 646 กำหนด National Code ไว้ 8 ช่อง แต่ภาษาไทยมีช่องทำได้เพียง 7 ช่องที่เหลือนี้ จึงกำหนดรหัสที่ใช้น้อยสุดไว้ที่ FB คือ โคมูตร เป็นต้น

เหตุผลของการเว้นช่องที่ซ้อนเพจกับ ISO 646 ตรงส่วน National Code ไว้ เพราะ

1. จากโอเอสของเครื่องหลายเครื่องที่ใช้ได้กับเครื่อง 7 บิทใช้รหัสที่ทำนี้เป็นรหัสควบคุม(กรณี 7 บิท) เช่น ใน XENIX และ UNIX (ดูจากหนังสือ XENIX Operating System for IBM PC ในเรื่องของ path และ pipe และหนังสือ The Business guide to XENIX System โดย Jean L Yates จะเห็นว่ารหัส 7 บิทดังกล่าวเป็นรหัสควบคุม)

นอกจากนี้ ในหนังสือ Inside Apple's Prodos โดย John CAMPBELL สำนักพิมพ์ Reston ได้ทำให้มองเห็นว่า PRODOS ของแอปเปิ้ลซึ่งเป็นดอสรุ่นใหม่ได้ใช้รหัสควบคุมโดยมีผลต่อภาษาไทยแน่นอน ดูบทว่าด้วย Machine Language interface

2. ในหนังสือ Programming Guide To the IBM PC โดย Peter Norton สำนักพิมพ์ Microsoft press ได้กล่าวไว้ในหน้า 412 ถึงหลักการ ON บิท 8 เพื่อใช้เป็นรหัส Software control ในโปรแกรมสำเร็จรูปต่างๆ โดยหลักการแล้วพบว่ามีทางเป็นไปได้ที่รหัส Softcode นี้จะอยู่ในพื้นที่ดังกล่าวโดยเฉพาะในงานเวิร์ดโปรเซสเซอร์หรือโปรแกรมประยุกต์เฉพาะที่ออกแบบในสหรัฐอเมริกา

3. จากหนังสือ Computer Network โดย Andrew S.Tanenbaun สำนักพิมพ์ Prentice Hall, 1981 หน้า 167–172 ได้กล่าวถึง Protocal SDLC/HDCC ซึ่งใช้รหัส 1 ติดกันหกบิทเป็นตัวบอกการเริ่มต้น ซึ่งในหนังสือเน้นว่าข้อมูลจะมีบิท 1 ติดกัน 6 บิทไม่ได้ จึงต้องมีขบวนการการแก้ไขและแสดงอัลกอริทึมการตรวจสอบไว้

4. ในระบบโอเอสภาษาคอมพิวเตอร์ยุคใหม่ และโลคัลแอเรียเนทเวอร์คอีกหลายระบบที่ต้องใช้รหัสควบคุมระบบพิเศษ ดังนั้นการเปิดช่องส่วนนั้นน่าจะปลอดภัยที่สุด เช่นในภาษาซีก็ใช้เป็นรหัสเอสเคป เป็นต้น

5. ฯลฯ

อนึ่งผู้เขียนเคยสนทนากับผู้เชี่ยวชาญต่างประเทศทั้งทางด้านญี่ปุ่นและอเมริกาต่างมีความเห็นว่าเพื่อลดความเสี่ยงในอนาคตน่าเว้นว่างในส่วน DB-DE และ FB-FE ไว้ โดยให้ยึดเสมือนเป็น National Code หรือรหัสที่จะได้รับการนำมาใช้ตามความต้องการในภายหลัง เพราะเราต้องยอมรับว่าเราไม่ได้เป็นผู้คิดระบบคอมพิวเตอร์ขึ้นเอง

รหัสภาษาไทยกับการประมวลผลจัดระดับและจัดเรียง (sort)

ในที่นี้ขอเสนออัลกอริทึมในการทำงานประมวลผลตัวอักษรไทย ให้เห็นว่ารหัสมีผลต่อการจัดระดับและจัดเรียงอย่างไร โดยเราต้องเข้าใจโครงสร้างของภาษาไทยก่อน(ผู้เขียนคิดว่าเราเข้าใจกันดีแล้ว) จะเห็นว่าไม่ว่าจะจัดเรียงลงช่องอย่างไรและนำมาประมวลผลตัวอักษรไทยจะต้องแยกเป็น 3 กลุ่ม เช่นถ้าจัดระดับก็แยกเป็นกลุ่มสระบนและวรรณยุกต์ กลุ่มสระล่าง กลุ่มพยัญชนะ ถ้านำมาจัดเรียงก็ต้องแยกกลุ่มตามหลักการของพจนานุกรมไทย

ดังนั้นก่อนที่จะเข้าสู่อัลกอริทึมจึงขอให้มาดูหลักการ Hashing และ IF...THEN

Hashing คืออะไร

Hashing เป็นวิธีการในการหยิบหรือหาข้อมูลอย่างรวดเร็วในลักษณะแรนดอมแอกเซส เพื่อว่าเวลาการค้นหาข้อมูลหรือหยิบข้อมูลจะไม่ขึ้นกับตำแหน่งที่อยู่ของข้อมูลเหมือนการแอกเซส RAM หรือแรนดอมไฟล์ ด้วยหลักการนี้เองทำให้การเข้าถึงรหัสไม่ขึ้นกับตำแหน่ง

ผู้สนใจอาจหาอ่านเพิ่มเติมจากหนังสือ "โครงสร้างข้อมูลเพื่อการออกแบบโปรแกรมคอมพิวเตอร์" ของดร.สุชาย ธวนเสถียร สำนักพิมพ์ซีเอ็ด หน้า 117 ผู้เขียนจะให้เห็นว่าทฤษฎี hashing จะทำให้ตัวอักษรไทยไม่มีปัญหาเหมือนที่หลายคนกังวล

เพื่อให้เข้าใจในหลักการ ขอยกตัวอย่างการใช้รหัสภาษาไทยด้านขบวนการการประมวลผลตามหลักวิชาการ

ในที่นี้จะสมมติว่าเราได้ตัวอักษรมาตัวหนึ่งจากคีย์บอร์ดจะนำไปแสดงผล รหัสที่จะส่งไปบนจอนี้ อาจเป็นสระบนหรือวรรณยุกต์สระล่างหรือพยัญชนะ วิธีการที่จะดูว่าอยู่ระดับใดนี้ เราใช้หลักการของ hashing ได้โดยตรงโดยไม่ต้องกังวลกับตำแหน่ง ลักษณะนี้เป็นการเปิดตารางนั่นเอง ลองดูในการเขียนเป็นแอสเซมบลี Z80 ของซีพีเอ็ม 8 บิทดู

```
LD  A, ข้อมูล      ; ข้อมูลที่รับมา
LD  B,A            ; เก็บสำรองไว้ที่ B
LD  HL, pointer    ; พอน์ยเตอร์ของตาราง
ADD  A,L
LD  L,A
LD  A, (HL)        ; รับรู้ค่าระดับของตัวอักษร
```

จะเห็นว่าเพียง 6 คำสั่งเท่านั้นเราก็รู้ว่าอักษรนั้นเป็นอะไรโดยมีตารางคอยบอก ซึ่งเห็นว่าช่วงเวลาการเข้าสู่ตารางไม่ขึ้นกับตำแหน่งที่อยู่ในตาราง

แต่ถ้าใช้หลักการของ IF จะต้องใช้คำสั่งมากกว่าแน่นอน ลองดูอัลกอริทึมการแสดงผลบน CRT ด้วย IF เขียนในรูปคำสั่งลำลอง

```
        Read ข้อมูล
    IF ข้อมูล  >  Upper
        Set Upper
    ELSE
        IF ข้อมูล  >  NORMAL
            set Normal
        ELSE
            set lower
        ENDIF
    ENDIF
```

จะเห็นว่าถ้าจัดข้อมูลตัวอักษรแยกตามกลุ่ม Upper, Normal และ Lower จะต้องใช้การเปรียบเทียบอย่างน้อย 2 ครั้ง วิธีนี้จะเกี่ยวพันโดยตรงกับการจัดวางตัวอักษรลงในตาราง และจะเห็นว่าวิธี Hashing ไม่มีผลต่อตำแหน่งของการจัดเรียงลงช่องเลย แต่ถ้าเราใช้วิธี IF การจัดตัว

อักษรลงในช่องจะมีผลอย่างยิ่งต่อจำนวนคำสั่ง IF และจำนวน IF ที่น้อยที่สุดจะเป็น 2 ครั้ง แต่การจัดทั่วไปจะให้ได้น้อยสุด 2 ครั้งได้ยาก เพราะจะต้องคำนึงถึงการจัดเรียงคำตามพจนานุกรมด้วย

อย่างไรก็ดี Hashing ต้องใช้หน่วยความจำเป็นตารางสำหรับเก็บข้อมูล ซึ่งทำให้ต้องใช้หน่วยความจำมากขึ้นเท่ากับจำนวนตาราง แต่ในปัจจุบันเกือบไม่มีปัญหาเรื่องหน่วยความจำ

การ Hashing ทำได้ง่ายและเร็วขึ้นเมื่อใช้แอสเซมบลี 8086

จากหนังสือโปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี 8086/8088 โดย รศ.ยืน ภู่วรวรรณ และ อ.สุรศักดิ์ สงวนพงษ์ สำนักพิมพ์ บริษัทซีเอ็ด หน้า 35 มีตัวอย่างการแปลงรหัส เช่น จาก EBCDIC เป็น ASCII ซึ่งเห็นว่ารหัส 2 กลุ่มนี้ไม่ลงตัวกันเลย แต่เราเขียนด้วยแอสเซมบลี 8086 ได้ ดังนี้

```
LEA BX, TABLE ; ชี้ไปตารางแปลงรหัส
MOV AL, ข้อมูล
XLAT
```

ก็เท่ากับเป็นการเอาข้อมูลไปแปลงเป็นรหัสตามที่ต้องการด้วยคำสั่งเพียง 3 คำสั่งเท่านั้น ดังนั้นผู้รู้จักใช้เทคนิคเหล่านี้จะพบว่า ไม่ว่าตารางรหัสเป็นอย่างไรก็ไม่มีผลกระทบต่อการประมวลผล แต่อย่างไรก็ตามรูปแบบของตารางสมอ.ก็ทำให้ใช้ IF ได้ไม่ยากเช่นกัน ดังจะเห็นได้จากตัว ° (นฤคหิต) และ (พินทุ) จะอยู่ในกลุ่มเพื่อให้ใช้ IF จัดระดับง่ายนั่นเอง และนฤคหิตกับพินทุเมื่อดูจากหนังสือพจนานุกรม ฉบับราชบัณฑิตยสถาน ปีพ.ศ. 2525 ไม่ได้กำหนดลำดับไว้ในหนังสือนั้น

อัลกอริทึมการแสดงผลบนจอภาพแบบสามระดับ

ในตารางของสมอ.ยึดหลักเป็นรหัสแลกเปลี่ยนข้อมูลและข่าวสารระหว่างกันเป็นจุดประสงค์แรก ดังนั้นจึงตัดเอารหัสแสดงผลที่เครื่องไมโครคอมพิวเตอร์จำนวนมากต้องการออกห้องปฏิบัติการวิจัยไมโครได้พัฒนาระบบภาษาไทยให้กับบริษัทต่างๆ หลายบริษัท โดยแต่ละบริษัทก็มีรหัสของตัวเองมาแล้ว และไม่ว่าจะเป็นรหัสแบบใดอัลกอริทึมในการแสดงผลก็คล้ายกัน เช่นในเครื่องแอปเปิ้ลมี TBASE 1, TBASE 2 และ TBASE 3 ซึ่งก็คือ Thai.com ที่ต่างรหัสกันนั่นเอง

ในอัลกอริทึมนี้ต้องการแสดงผลบนจอภาพ โดยจะทำการตรวจสอบลักษณะของตัวอักขระที่แสดงผล การรับตัวอักษรจะทำการรันมาทีละตัว ดังนั้นเพื่อให้แสดงผลในรูปแบบสามระดับจึงต้องมีการสร้างสถานะ flag เพื่อจะบอกให้ทราบว่าตัวก่อนวรรณยุกต์ ่ ้ ๊ ๋ ์ เป็นสระที่จะรวมได้หรือไม่ โดยสถานะของแฟล็กจะเป็นตัวบอกให้ไปหาตัวรหัสแสดงผลแบบผสม (สมอ.ไม่ได้กำหนด) อัลกอริทึมนี้จึงขึ้นกับวิธีการหาตัวแสดงผลแบบผสมนี้ด้วย

อัลกอริทึมดังกล่าวนี้แสดงได้ดังรูปที่ 6 (อ้างอิงโดยใช้รหัสเกษตรฯ)

เพื่อให้ผู้อ่านได้นำเอาไปใช้จริงได้ จึงนำวิธีการนี้มาเขียนด้วยภาษาเบสิกดังรูปที่ 7 โดยใช้ MSBASIC และใช้ device driver คือ ANSI.SYS เพื่อกำหนด Screen Command ดังนั้นเมื่อจะบูทระบบต้องมี

รูปที่ 6 อัลกอริทึมการจัดเรียงตัวอักษรบนจอ CRT (อัลกอริทึม DT)

```
PROCEDURE thai - crt - display
    clear flag: = 0;
    locate cursor: x,y;
    input character;
    IF character: = normal ( < 215)
        display;
        forward cursor;
        clear flag: = 0
    ELSE
        IF character: = lower ( < 217)
            back space and down cursor
            display
            up cursor and forward cursor
            clear flag: = 0
        ELSE
            IF character: = ่,้,๊,๋,์
                IF flag: ≠ 0
                    find combine (depend on flag)
                    back space and up cursor
                    display
                    down cursor and forward cursor
                    clear flag: = 0
                ELSE
```

```
                    back space and up cursor
                    display
                    down cursor and forward cursor
                    set flag: = weight
                ENDIF
            ENDIF
        END IF
    END IF
END thai - ort - display
```

รูปที่ 7 โปรแกรมป้อนข้อมูลแบบ 3 ระดับ

```
10 DEFINT A-Z
20 DIM A(127),U(30),N(30),L(30)
30 DEF FNCUR$(Y,X)=CHR$(27)+"["+RIGHT$(STR$(Y+1),LEN(STR$(Y+1))-1)+";"+RIGHT$(ST
R$(X+1),LEN(STR$(X+1))-1)+"f"
40 AT$=CHR$(27)+"[s":BT$=CHR$(27)+"[u":CLEARSCREEN$=CHR$(27)+"[2J"
50 CURUP$=CHR$(27)+"[A":CURDOWN$=CHR$(27)+"[B":CURLF$=CHR$(27)+"[D":DEL$=" "+CHR
$(8)
60 GOSUB 880:U(0)=32:N(0)=32:L(0)=32
70 PRINT CLEARSCREEN$:PRINT FNCUR$(3,20);"Thai input for Thai conin routine"
80  PRINT FNCUR$(5,20);"Thai word: ";:GOSUB 230:PRINT:PRINT:PRINT
90 IF VN$="" THEN PRINT CLEARSCREEN$:END
100 REM
110 PRINT
120 NNN=LEN(VN$)
130 RES$ = ""
140 PRINT FNCUR$(12,20);"translated into one line"
150 FOR II = 1 TO NNN
160 RES$ = RES$+MID$(VN$,II,1)
170 IF MID$(VL$,II,1)<>" " THEN RES$=RES$+MID$(VL$,II,1)
180 IF MID$(VU$,II,1)<> " " THEN RES$=RES$+MID$(VU$,II,1)
190 NEXT II
200 PRINT FNCUR$(15,20);RES$
210 X$=INPUT$(1)
220 GOTO 70
230 '
240 '    INPUT CHARACTER
250 '
260 C=1:VU$="":VN$="":VL$=""
270 CONIN$=INPUT$(1):GOSUB 280:IF CHAR=13 OR C>30 THEN RETURN ELSE GOTO 270
280 PRINT AT$;FNCUR$(0,0):PRINT BT$;
290 CHAR=A(ASC(CONIN$))
300 IF CHAR<32 OR CHAR=127 THEN 810
310 '
320 '    NORMAL
330 '
340 IF CHAR>214 THEN 460
350 GOSUB 360:GOTO 420
360   COUNTC=LEN(VN$)-C:COUNTC1=C-1:C=C+1
370   IF COUNTC<0 THEN COUNTC=0
380   VU$=VU$+" "
390   VN$=VN$+CHR$(CHAR)
400   VL$=VL$+" "
410 FLAG=0:K=0:I=0:RETURN
```

```
420 PRINT CHR$(CHAR);CURUP$;CURLF$;" ";CURDOWN$;CURDOWN$;CURLF$;" ";CURUP$;:RETU
RN
430 '
440 '    LOWER
450 '
460 '
470 IF CHAR=215 OR CHAR=216 THEN 480 ELSE 540
480 IF C=1 THEN RETURN
490 VL$=LEFT$(VL$,C-2)+CHR$(CHAR)
500 PRINT CURDOWN$;CURLF$;CHR$(CHAR);CURUP$;:RETURN
510 '
520 '    UPPER
530 '
540 IF CHAR>223 THEN 650
550 IF C=1 THEN RETURN ELSE GOSUB 560:GOTO 600
560 IF CHAR=223 THEN FLAG=0:RETURN
570 I=CHAR-216:IF K=0 THEN FLAG=0
580 IF K=5 AND I<>1 THEN FLAG=0
590 IF FLAG=0 THEN FLAG=1:K=0:RETURN ELSE 700
600 VU$=LEFT$(VU$,C-2)+CHR$(CHAR)
610 PRINT CURUP$;CURLF$;CHR$(CHAR);CURDOWN$;:RETURN
620 '
630 '    SUPER
640 '
650 IF C=1 THEN RETURN ELSE GOSUB 660:GOTO 600
660 IF CHAR=235 THEN I=5:K=2:FLAG=1:RETURN
670 IF CHAR=228 AND I<>1 THEN K=5:FLAG=1:RETURN
680 K=CHAR-223:IF I=0 THEN FLAG=0
690 IF FLAG=0 THEN FLAG=1:I=0:RETURN
700 ON I GOSUB 720,730,740,750,760,770
710 CHAR=Y:RETURN
720 Y=K+237:RETURN
730 Y=K+242:RETURN
740 Y=K+246:RETURN
750 Y=K+250:RETURN
760 Y=K+233:RETURN
770 Y=K+229:RETURN
780 '
790 '    CONTROL CHARACTER
800 '
810 IF CHAR=8 THEN GOSUB 830
820 FLAG=0:K=0:I=0:RETURN
830 IF C=1 THEN RETURN
840 VU$=LEFT$(VU$,LEN(VU$)-1)
850 VN$=LEFT$(VN$,LEN(VN$)-1)
860 VL$=LEFT$(VL$,LEN(VL$)-1)
870 PRINT CURLF$;DEL$;CURDOWN$;DEL$;CURUP$;CURUP$;DEL$;CURDOWN$;:C=C-1:RETURN
880 '
890 '    READ TABLE
900 '
910 FOR J=1 TO 127
920     READ A(J)
930 NEXT J
940 RETURN
950 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
960 DATA 19,20,21,22,23,24,25,26,27,28,29,30,31,32,46
970 DATA 95,50,51,52,235,165,54,55,53,57,191,162,211
980 DATA 187,166,195,45,47,190,180,215,219,163,179
990 DATA 169,196,176,168,201,32,49,194,63,167,173
1000 DATA 172,210,170,223,177,227,198,197,44,228,214
1010 DATA 171,48,175,164,182,226,203,34,41,32,40,184
1020 DATA 32,195,216,56,32,189,217,209,161,207,178
1030 DATA 208,225,193,224,206,199,181,220,183,192,213
1040 DATA 188,200,204,218,202,212,185,221,186,174,32
1050 DATA 44,32,127
```

ANSI.SYS อยู่ในดิสค์และมีโปรแกรม CONFIG.SYS โดยในไฟล์ CONFIG.SYS มีข้อความว่า DEVICE = ANSI.SYS

โปรแกรมนี้เป็นการกำหนดตำแหน่งเคอร์เซอร์ไปที่ใดก็ได้บนจอภาพ แล้วป้อนข้อมูลแบบ 3 ระดับภาษาไทยบนจอภาพ เมื่อป้อนเสร็จกดรีเทอร์น โปรแกรมจะแปรข้อมูลกลับเป็นแบบระดับเดียวอักษรพ่อขุนราม–คำแหง แล้วพิมพ์ออกมาบนจอให้ดู

ผู้ใช้จึงนำโปรแกรมนี้ไปเขียนโปรแกรมประยุกต์อื่นๆ ได้มาก ลองดูสิครับแล้วคุณจะเห็นว่าเล่นกับภาษาไทยไม่ยากถ้าได้ลองลงมือทำดู แล้วจะพบว่าภาษาไทยเป็นภาษาที่มีลักษณะเฉพาะตัว ที่เราสามารถนำคอมพิวเตอร์มาประยุกต์ได้เสมอ

สิ่งที่น่าสังเกตจากโปรแกรมนี้คือ เราใช้ ANSI ซึ่งเป็นมาตรฐานของคำสั่งจัดจอภาพของเทอร์มินัลที่ใช้กัน โดยบรรทัดที่ 20 กำหนดอะเรย์ A(127) เพื่อไว้สำหรับทำ Hashing การรันคีย์เพื่อเปลี่ยนเป็นไทย โดยบรรทัดที่ 950–1050 เป็นตารางรหัสภาษาไทย ในที่นี้ขอใช้รหัสแสดงผลที่มีอักษรผสม จึงขอใช้รหัสเกษตรซึ่งเป็น display code หากใช้ของสมอ.จะต้องแสดงในรูปแบบ 4 บรรทัดอาจไม่สวย แต่โปรแกรมก็ทำได้ไม่ยาก

บรรทัดที่ 30 เป็นการสร้างฟังก์ชันกำหนดจอภาพ เช่น กำหนดตำแหน่งเคอร์–เซอร์ กำหนดการเลื่อนเคอร์เซอร์เพื่อจะได้นำไปใช้ในการจัดตัวอักษรไทย อัลกอริทึมนี้ใช้การแบ่งเป็นสามระดับโดยแสดงผล และการลบข้อมูลเมื่อถอยเคอร์เซอร์กลับได้อย่างดี โปรแกรมอาจเขียนยาวไปหน่อย แต่ก็มีขีดความสามารถที่จะเชื่อมโยงกับโปรแกรมทางธุรกิจได้ไม่ยาก

เมื่อรันโปรแกรมนี้จะได้ผลบนจอให้เราป้อนข้อมูล ครั้นเมื่อคีย์เข้าภาษาไทยจะได้รับการจัดแบบถูกอักขระ โดยสามารถเลื่อนถอยหลังมาลบได้ โปรแกรมกำหนดคอลัมน์การแสดงผลเพียง 30 คอลัมน์ เมื่อกดรีเทอร์นเลิกป้อนข้อมูล โปรแกรมบรรทัดที่ 100–220 จะนำข้อมูลจากตัวแปร VN$, VL$, VU$ มาแปลงเป็น RES$ ผลของบนจอจะเป็นดังรูปที่ 8

อัลกอริทึมการจัดพิมพ์

ในการจัดพิมพ์ตัวอักษรบนเครื่องพิมพ์จะมีลักษณะบางอย่างที่แตกต่างจากการแสดงผลบนจอภาพ เพราะเราจะต้องนำสตริงมาแตกเป็น 3 ระดับ แล้วจึงสั่งให้เครื่องพิมพ์ๆ ทีละบรรทัด โดยเริ่มจากพิมพ์สระบนก่อนแล้วพิมพ์ตัวกลาง พิมพ์สระล่าง โดยในที่นี้สมมติให้ Y9$ คือสตริงที่ต้องการจะพิมพ์ โดยอัลกอริทึมนี้จะแยกออกเป็น Y6$, Y7$ และ Y8$ เพื่อแสดงผลการพิมพ์ในระดับต่างๆ กัน

อัลกอริทึมการพิมพ์ผลภาษาไทยบนเครื่องพิมพ์ในที่นี้ข้อกำหนดคือเป็น PT ดังรูปที่ 9

ในที่นี้จะขอเขียนด้วยเบสิกให้ดูอีก ความจริงน่าจะเขียนด้วยปาสคาล แต่เนื่องจากไมโครได้เคยเสนอแล้วในบทความของอาจารย์บุญเลิศ เลยขอถือโอกาสเขียนด้วยเบสิกให้ดู เพื่อว่าจะเห็นประโยชน์และนำไปใช้ได้ แม้จะอยู่ในภาษาเบสิกก็ทำงานได้ดี

รูปที่ 8 ผลที่ได้จากการรันโปรแกรม Thai conin

```
Thai input for Thai conin routine
                 ั  ส  ั    ส่ั
Thai word: สวสดครบคณหรก  <------- ป้อนข้อมูลเข้า
                    ่

translated into one line

สวัสดีครับคุณหรัก  <------- แปลงเป็นระดับเดียว
```

รูปที่ 9 อัลกอริทึม PT

```
PROCEDURE thai - print
     set i = 1
     set j = 2
     DOWHILE  .not. end of string
          char $ = MID(Y9$, i, 1)
          IF char $ ≠ normal ( > 214)
               IF char $ =_ _ _ _ _ . . ( < 224.and. > 216)
                    push on Y6$ at position j - 1
                    set flag (j - 1) = weight
               ELSE
                    IF char $ =. .  ( < 217)
```

```
                    push on Y8$ at position j - 1
                ELSE
                    IF char $ =' ' ' ' ' ' ( < 229)
                        IF flag (j - 1) = weight ( > 0)
                            Do combine char
                        END IF
                        push on Y6$ at position j
                        set flag (j - 1) = 0
                    END IF
                END IF
            END IF
        ELSE
            push on Y7$ at position j
            j = j + 1
        END IF
        i = i + 1
    END DO
    Print Y6$
    Print Y7$
    Print Y8$
END thai - print
```

รูปที่ 10 โปรแกรมพิมพ์ผลแบบ 3 ระดับ

```
1900 'thai print routine
1910 'we accept thai string (variable = res$)
1920 'one line is in res$
1930 'we convert into three line
1940 '
2000 LL$=" "
2010 LN$=" "
2020 LU$=" "
2030 COUNT = LEN(RES$)
2034 IF COUNT = 0 THEN GOTO 2120
2040 FOR I = 1 TO COUNT
2050 IF ASC(MID$(RES$,I,1))>=217 THEN LU$=LEFT$(LU$
,LEN(LU$)-1)+MID$(RES$,I,1):GOTO 2080
2060   IF ASC(MID$(RES$,I,1))<215 THEN LN$=LN$+MID$
(RES$,I,1):LL$=LL$+" ":LU$=LU$+" ":GOTO 2080
2070 LL$=LEFT$(LL$,LEN(LL$)-1)+MID$(RES$,I,1)
2080 NEXT I
2090 LPRINT LU$
2100 LPRINT LN$
2110 LPRINT LL$
```

สวัสดีครับคุณที่รัก ← เมื่อรันจะได้ผลลัพธ์ดังนี้

โปรแกรมที่เสนอในรูปที่ 10 นี้ ส่งสตริง RES$ = "สวัสดีครับคุณที่รัก" มาให้ เราจะแยกออกเป็น LU$ LN$ และ LL$ ซึ่งจะเห็นได้ว่าเราเขียนโปรแกรมได้ไม่ยากเลย

สังเกตว่าอัลกอริทึมการแสดงผลบนจอและการพิมพ์บนเครื่องพิมพ์นี้ ถ้าใช้ IF จะต้องมีการจัดให้ตรวจสอบได้ไม่ยาก รหัสของเกษตรยึดหลักการเป็นรหัส display และใช้เงื่อนไขการแบ่งระดับเป็นจุดใหญ่ เพราะสำคัญในการกรอกลงช่อง แต่ถึงแม้จะใช้กับรหัสสมอ.ซึ่งจะต้องมีการใช้ IF มากกว่า ก็ไม่ได้หมายความว่ารหัสสมอ.ใช้ยาก ซึ่งความจริงแล้วก็เพิ่ม IF อีกเล็กน้อยเท่านั้น หรือถ้าเขียนด้วยแอสเซมบลีจะใช้ Hashing อย่างนี้นำเสนอ ก็จะทำให้ได้เร็วไม่แพ้การจัดรหัสแบบอื่น

ในตอนต่อไปจะได้กล่าวถึง การจัดเรียงคำไทย ซึ่งมีผลต่อการวางตำแหน่งตัวอักษรลงในตาราง และอัลกอริทึมที่ใช้ในการจัดเรียงคำไทย

หมายเหตุ ปัจจุบัน THAI.COM ที่ออกแบบสร้างขึ้นนั้น ก็คือ CONIN, CONOUT และ PRINT เหมือนที่เสนอนี้ แต่เขียนในรูปภาษาแอสเซมบลีและเชื่อมโยงกับ MSDOS (ดูในวารสารไมโคร เล่ม 21)

แก้ที่ผิดไมโครฯ ฉบับที่ 22

เรื่อง

เทคนิคพัฒนาการภาษาไทย
บนเครื่องไมโครคอมพิวเตอร์
ตอน 3

- หน้า 70 และ 71 สลับหน้ากัน
- หน้า 72 และ 73 สลับหน้ากัน
- หน้า 78 ตารางซ้ายด้านบนเป็นส่วนต่อของตารางที่ 6 และตารางซ้ายด้านล่างและตารางขวา เป็นตารางที่ 7 *(ตารางแสดงการแจกแจงตัวอักษรที่ใช้เป็นตัวสุดท้าย)*